# Parameter-Free Spatial Data Mining Using MDL

Spiros Papadimitriou[§ *] Aristides Gionis[‡] Panayiotis Tsaparas[‡]
Risto A. Väisänen[‡] Heikki Mannila[‡] Christos Faloutsos[§ †]

[§] Carnegie Mellon University
Pittsburgh, PA, USA

[‡] University of Helsinki
Helsinki, Finland

## Abstract

*Consider spatial data consisting of a set of* binary features *taking values over a collection of* spatial extents *(grid cells). We propose a method that simultaneously finds spatial correlation and feature co-occurrence patterns, without* any *parameters. In particular, we employ the Minimum Description Length (MDL) principle coupled with a natural way of compressing regions. This defines what "good" means: a feature co-occurrence pattern is good, if it helps us better compress the set of locations for these features. Conversely, a spatial correlation is good, if it helps us better compress the set of features in the corresponding region. Our approach is scalable for large datasets (both number of locations and of features). We evaluate our method on both real and synthetic datasets.*

## 1 Introduction

In this paper we deal with the problem of finding spatial correlation patterns and feature co-occurrence patterns, *simultaneously* and *automatically*. For example, consider environmental data where spatial locations correspond to patches (cells in a rectangular grid) and features correspond to species presence information. For each patch and species pair, the observed value is either one or zero, depending on whether the particular species was observed or not at that patch. In this case, feature co-occurrence patterns would correspond to species co-habitation and spatial correlation patterns would correspond to natural habitats for species groups. Combining the two will generate homogeneous regions characterised by a set of species that live in those regions. We wish to find "good" patterns of this form *simultaneously* and *automatically*.

Spatial data in this form (binary features over a set of locations) occur naturally in several settings, e.g.:

- Biodiversity data, such as the example above.
- Geographical data, e.g., presence of facilities (shops, hospitals, houses, offices, etc) over a set of city blocks.
- Environmental data, e.g., occurrence of different phenomena (storms, hurricanes, snow, drought, etc. or ) over a set of locations in satellite images.
- Historical/linguistic data, e.g., occurrence of different words in different counties, or occurrence of various types of historical events over a set of locations.

In all these settings, we would like to discover meaningful feature co-occurrence and spatial correlation patterns. Existing methods either discover one of the two types of patterns in isolation, or require the user to specify certain parameters or thresholds.

We view the problem from the perspective of succinctly summarizing (i.e., compressing) the data, and we employ the Minimum Description Length (MDL) principle to automate the process. We group locations and features *simultaneously*: feature co-occurrence patterns help us compress spatial correlation patterns better, and vice versa. Furthermore, for location groups, we incorporate spatial affinity by compressing regions in a natural way.

Section 2 presents some of the background, in the context of our problem. Section 3 builds upon this background, leading to the proposed approach described in Section 4. Section 5 presents experiments that illustrate the results of our approach. Section 6 surveys related work. Finally, in Section 7 we conclude.

## 2 Background

In this section we introduce some background, in the context of the problem we wish to solve. In subsequent sections we explain how we adapt these techniques for our purposes.

### 2.1 Minimum description length (MDL)

In this section we give a brief overview of a practical formulation of the minimum description length (MDL) principle. For further information see, e.g., [5, 8]. Intuitively,

[*] Part of this work was done while the author was visiting Basic Research Unit, HIIT, University of Helsinki, Finland.

[†] This material is based upon work supported by the National Science Foundation under Grants No. IIS-0083148, IIS-0209107, IIS-0205224, INT-0318547, SENSOR-0329549, EF-0331657, IIS-0326322, NASA Grant AIST-QRS-04-3031, CNS-0433540. This work is supported in part by the Pennsylvania Infrastructure Technology Alliance (PITA). Additional funding was provided by donations from Intel, and by a gift from Northrop-Grumman Corporation. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation, or other funding parties.

the main idea behind MDL is the following: Let us assume that we have a family $\mathscr{M}$ of *models* with varying degrees of complexity. More complex models $M \in \mathscr{M}$ involve more parameters but, *given* these parameters (i.e., the model $M \in \mathscr{M}$), we can describe the observed data more concisely.

As a simple, concrete example, consider a binary sequence $D := [d(1), d(2), \ldots, d(n)]$ of $n$ coin tosses. A simple model $M^{(1)}$ might consist of specifying the number $h$ of heads. Given this model $M^{(1)} \equiv \{h/n\}$, we can encode the dataset $D$ using $L(D|M^{(1)}) := nH(h/n)$ bits [26], where $H(\cdot)$ is the Shannon entropy function. However, in order to be fair, we should also include the number $L(M^{(1)})$ of bits to transmit the fraction $h/n$, which can be done using $\log^\star n$ bits for the denominator and $\lceil \log(n+1) \rceil$ bits for the numerator $h \in \{0, 1, \ldots, n\}$, for a total of $L(M^{(1)}) := \log^\star n + \lceil \log(n+1) \rceil$ bits.

**Definition 1 (Code length and description complexity)** $L(D|M^{(1)})$ *is* code length *for D, given the model* $M^{(1)}$. $L(M^{(1)})$ *is the* model description complexity *and* $L(D, M^{(1)}) := L(D|M^{(1)}) + L(M^{(1)})$ *is the* total code length.

A slightly more complex model might consist of segmenting the sequence in two pieces of length $n_1 \geq 1$ and $n_2 = n - n_1$ and describing each one independently. Let $h_1$ and $h_2$ be the number of heads in each segment. Then, to describe the model $M^{(2)} \equiv \{h_1/n_1, h_2/n_2\}$, we need $L(M^{(2)}) := \log^\star n + \lceil \log n \rceil + \lceil \log(n - n_1) \rceil + \lceil \log(n_1 + 1) \rceil + \lceil \log(n_2 + 1) \rceil$ bits. Given this information, we can describe the sequence using $L(D|M^{(2)}) := n_1 H(h_1/n_1) + n_2 H(h_2/n_2)$ bits.

Now, assume that our family of models is $\mathscr{M} := \{M^{(1)}, M^{(2)}\}$ and we wish to choose the "best" one for a particular sequence $D$. We will examine two sequences of length $n = 16$, both with 8 zeros and 8 ones, to illustrate the intuition.

Let $D_1 := \{0, 1, 0, 1, \cdots, 0, 1\}$, with alternating values. We have $L(D_1|M_1^{(1)}) = 16H(1/2) = 16$ and $L(M_1^{(1)}) = \log^\star 16 + \lceil \log(16+1) \rceil = 10 + 5 = 15$. However, for $M_1^{(2)}$ the best choice is $n_1 = 15$, with $L(D_1|M_1^{(2)}) \approx 15$ and $L(M_1^{(2)}) \approx 19$. The *total* code lengths are $L(D_1, M_1^{(1)}) \approx 16 + 15 = 31$ and $L(D_1, M_1^{(2)}) \approx 15 + 19 = 34$. Thus, based on total code length, the simpler model is better[1]. The more complex model may give us a lower code length, but that benefit is not enough to overcome the increase in description complexity: $D_1$ does not exhibit a pattern that can be exploited *by a two-segment model* to describe the data.

Let $D_2 := \{0, \cdots, 0, 1, \cdots, 1\}$ with all similar values contiguous. We have again $L(D_2|M_2^{(1)}) = 16$ and $L(M_2^{(1)}) = 15$. But, for $M_2^{(2)}$ the best choice is $n_1 = n_2 = 8$ so that $L(D_2|M_2^{(2)}) = 8H(0) + 8H(1) = 0$ and $L(M_2^{(2)}) \approx 24$. The *total* code lengths are $L(D_2, M_2^{(1)}) \approx 16 + 15 = 31$ and $L(D_2, M_2^{(2)}) \approx 0 + 24 = 24$. Thus, based on total code length, the two-segment model is better. Intuitively, it is clear that $D_2$ exhibits a pattern that can help reduce the total code length. This intuitive fact is precisely captured by the total code length.

[1]The absolute codelengths are not important; the bit overhead compared to the straight transmission of $D$ tends to zero, as $n$ grows to infinity.

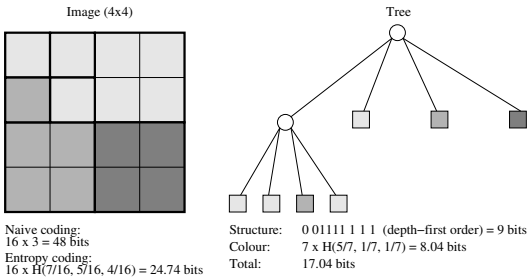



**Figure 1. Quadtree compression: The map on the left has $4 \times 4 = 16$ cells (pixels), each having one of three possible values. The resulting quadtree has 10 leaf nodes, again each having one of three possible values.**

In fact, this simple example is prototypical of the groupings we will consider later. More generally, we could consider a family $\mathscr{M} := \{M^{(k)} \mid 1 \leq k \leq n\}$ of $k$-segment models and apply the same principles. Furthermore, the datasets we will consider are two-dimensional matrices $D := [d(i, j)]$, instead of one-dimensional sequences. In Section 3.2 we address both of these issues. To complicate matters even further, one of the dimensions of $D$ has a spatial location associated with it. Section 4 presents data description models that also incorporate this information.

In fact, choosing the appropriate family of models is nontrivial. Roughly speaking, at one extreme we have the singleton family of "just the raw data," which cannot describe any patterns. At the other extreme, we have "all Turing machine programs that produce the data as output," which can describe the most intricate patterns, but make model selection intractable. Striking the right balance is a challenge. In this paper, we address it for the case of spatial data.

## 2.2 Quadtree compression

A quadtree is a data structure that can be used to efficiently index contiguous regions of variable size in a grid. It has been used successfully in image coding and has the benefit of small overhead and very efficient construction [28]. Figure 1 shows a simple example. Each internal node in a quadtree corresponds to a partitioning of a rectangular region into four quadrants. The leaf nodes of a quadtree represent rectangular groups of cells and have a value $p$ associated them, where $p$ is the group ID. In the following we briefly describe quadtree codelengths.

**Structure.** The structure of a quadtree uniquely corresponds to a partitioning of the grid. For example, the partitioning into three regions in Figure 1 on the left corresponds to the structure on the right. This partitioning is chosen in a way that respects spatial correlations. The structure can be described easily by performing a traversal of the tree and transmitting a zero for non-leaf nodes and a one for leaf nodes. The traversal order is not significant; we choose depth-first order (see Figure 1).

**Values.** Quadtree structure conveys information about the partition boundaries (thick grid lines in Figure 1). These capture all correlations: in effect, we have reduced the original set of equal-sized cells to a (smaller) set of variable-sized, square cells (each one corresponding to a leaf node in the quadtree). Since the correlations have already been taken into account, we may assume that the leaf node values are independent. Therefore, the cost to transmit the values is equal to the total number of leaf nodes, multiplied by the entropy of the leaf value distribution.

**Lemma 1 (Quadtree codelength)** *Let $T$ be a quadtree with $m'$ leaf nodes, of which $m'_p$ have value $p$, where $1 \leq p \leq k$. Then, the number of internal nodes is $\lceil m'/3 \rceil - 1$. Structure information can be transmitted using one bit per node (leaf/non-leaf) and values can be transmitted using entropy coding. Therefore, the corresponding total codelength is*

$$L(T) = m'H\left(\frac{m'_1}{m'}, \frac{m'_2}{m'}, \ldots, \frac{m'_k}{m'}\right) + \left\lceil \frac{4m'}{3} \right\rceil - 1$$

This has a straightforward but important consequence:

**Lemma 2** *The codelength $L(T)$ for a quadtree $T$ can be computed in constant time, if we know the distribution of leaf node values.*

In other words, for a full quadtree (i.e., one where each node has either zero or four descendants), if we know $m'$ and $m'_p$, for $1 \leq p \leq k$, we can compute the cost in closed form, using Lemma 1. Note that the quadtree does not have to be perfect (i.e., all leaves do not have to be at the same level). When a node is reassigned a different value, region consolidations may occur (i.e., pruning of leaves with same value). Updating $m'$ and $m'_p$ will require time proportional to the number of consolidations, which are typically localized. In the worst case, the time will be $O(\log m)$ if pruning cascades up to the root node.

# 3 Preliminaries

In this section we formalize the problem and prepare the ground for introducing our approach in Section 4.

## 3.1 Problem definition

Assume we are given $m$ cells on an evenly-spaced grid (e.g., field patches in biological data) and $n$ features (e.g., species). For each pair $(i, j)$, $1 \leq i \leq m$ and $1 \leq j \leq n$, we are also given a binary observation (e.g., species presence/absence at each cell).

| Symbol | Definition |
|---|---|
| $D$ | Binary data matrix. |
| $m, n$ | Dimensions of $D$ (rows, columns); rows correspond to cells. |
| $k, \ell$ | Number of row and column groups. |
| $k^*, \ell^*$ | Optimal number of groups. |
| $Q_X, Q_Y$ | Row and column assignments to groups. |
| $D_{p,q}$ | Submatrix for intersection of $p$-th row and $q$-th column group. |
| $m_p, n_q$ | Dimensions of $D_{p,q}$. |
| $\lvert D_{p,q}\rvert$ | Number of elements $\lvert D_{p,q}\rvert := m_p n_q$. |
| $\rho_{p,q}$ | Density of 1s in $D_{p,q}$. |
| $H(\cdot)$ | Binary Shannon entropy function. |
| $L(D_{p,q} \lvert Q_X, Q_Y, k, \ell)$ | Codelength for $D_{p,q}$. |
| $L(D, Q_X, Q_Y, k, \ell)$ | Total codelength for $D$. |

**Table 1. Symbols and definitions.**

We want to group both cells and features, thus also implicitly forming groups of observations (each such group corresponding to an intersection of cell and feature groups). The two main requirements are:

1. Spatial affinity: Groups of cells should exhibit spatial coherence, i.e., if two cells $i_1$ and $i_2$ are close together, then we wish to favour cell groupings that place them in the same group. Furthermore, spatial affinity should be balanced with feature affinity in a principled way.
2. Homogeneity: The implicit groups of observations should be as homogeneous as possible, i.e., be nearly all-ones or all-zeros.

The problem and our proposed solution can be easily extended to a collection of categorical features (i.e., taking more than two values, from a finite set of possible values) per cell.

## 3.2 MDL and binary matrices

Let $D = [d(i, j)]$ denote a $m \times n$ ($m, n \geq 1$) binary data matrix. A *bi-grouping* is a simultaneous grouping of the $m$ rows and $n$ columns into $k$ and $\ell$ disjoint row and column groups, respectively. Formally, let

$$Q_X : \{1, 2, \ldots, m\} \to \{1, 2, \ldots, k\}$$
$$Q_Y : \{1, 2, \ldots, n\} \to \{1, 2, \ldots, \ell\}$$

denote the assignments of rows to row groups and columns to column groups, respectively. The pair $\{Q_X, Q_Y\}$ is a bi-grouping.

Based on the observation that a good compression of the matrix implies a good, concise grouping, both $k$, $\ell$ as well as the assignments $Q_X, Q_Y$ can be determined by optimizing the description cost of the matrix. Let

$$R_p := Q_X^{-1}(p), \quad C_q := Q_Y^{-1}(q), \quad 1 \leq p \leq k, 1 \leq q \leq \ell$$

be the set of rows and columns assigned to row group $p$ and column group $q$, with sizes $m_p := |R_p|$ and $n_q := |C_q|$, respectively. Then, let

$$D_{p,q} := [d(R_p, C_q)], \quad 1 \le p \le k, 1 \le q \le \ell.$$

be the sub-matrix of $D$ defined by the intersection of row group $p$ and column group $q$. The total codelength $L(D) \equiv L(D, Q_X, Q_Y, k, \ell)$ for transmitting $D$ is expressed as

$$L(D) = L(D|Q_X, Q_Y, k, \ell) + L(Q_X, Q_Y, k, \ell).$$

For the first part of Eq. 3.2, elements within each $D_{p,q}$ are assumed to be drawn independently, so that

$$L(D_{p,q}|Q_X, Q_Y, k, \ell) = \lceil \log(|D_{p,q}| + 1) \rceil + |D_{p,q}| H(\rho_{p,q})$$

where $\rho_{p,q}$ is the density (i.e., probability) of ones within $D_{p,q}$ and $|D_{p,q}| = m_p n_q$ is the number of its elements. This is analogous to the coin toss sequence models described in Section 2.1. Finally,

$$L(D|Q_X, Q_Y, k, \ell) := \sum_{p=1}^{k} \sum_{q=1}^{\ell} L(D_{p,q}|Q_X, Q_Y, k, \ell).$$

For the second part of Eq. 3.2, row and column groupings are assumed to be independent, hence

$$\begin{aligned} L(Q_X, Q_Y, k, \ell) &= L(Q_X, Q_Y|k, \ell) + L(k, \ell) \\ &= L(Q_X|k) + L(k) + L(Q_Y|\ell) + L(\ell). \end{aligned}$$

Finally, a uniform prior is assigned to the number of groups, as well as to each possible grouping given the number of groups, i.e.,

$$\begin{aligned} L(k) &= -\log \Pr(k) = \log m \\ L(Q_X|k) &= -\log \Pr(Q_X|k) = \log \binom{m}{m_1 \cdots m_k} \end{aligned}$$

and similarly for the column groups.

Using Stirling's approximation $\ln n! \approx n \ln n - n$ and the fact that $\sum_i m_i = m$, we can easily derive the bound

$$\begin{aligned} L(Q_X|k) &= \log \binom{m}{m_1 \cdots m_k} = \log \frac{m!}{m_1! \cdots m_k!} \\ &= \log m! - \textstyle\sum_{i=1}^{k} \log m_i! \approx m \log m - \sum_{i=1}^{k} m_i \log m_i \\ &= -m \textstyle\sum_{i=1}^{k} \frac{m_i}{m} \log \frac{m_i}{m} = mH\left(\frac{m_1}{m}, \ldots, \frac{m_k}{m}\right) \\ &\le m \log k. \end{aligned}$$

Therefore, we have the following:

**Lemma 3** *The codelength for transmitting an arbitrary m-to-k mapping $Q_X$, where $m_p$ symbols from the range are mapped into each value $p, 1 \le p \le k$, is approximately*

$$L(Q_X|k) = mH\left(\tfrac{m_1}{m}, \ldots, \tfrac{m_k}{m}\right)$$

### 3.3 Map boundaries

The set of all cells may form an arbitrary, complex shape, rather than a square with a side that is a power of two. However, we wish to penalize only the complexity of interior cell group boundaries. The shape of boundaries on the edges (e.g., coastline) of the map should not affect the cost.

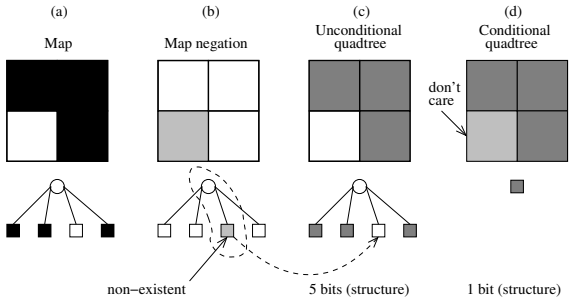



**Figure 2. Quadtree compression to discount the complexity of the enclosing region's shape; only the complexity of cell group shapes *within* the map's boundaries matters.**

For example, assume that our dataset consists of the three black cells in Figure 2(a). If all three cells belong to the same group and we encode this information naïvely, then we get a quadtree with five nodes (Figure 2(c)). However, the complexity of the resulting quadtree is only due to the fact that the bottom-left is "non-existent."

If we know the shape of the entire map *a priori*, we can encode the same information using 1 bit, as shown in Figure 2(d). In essence, both transmitter and receiver agree upon a set of "existing" cell locations (or, equivalently, a *prior* quadtree corresponding to the map description). This information should not be accounted for in the total codelength, as it is fixed for a given dataset. Given this information, all cells groups in the transmitted quadtree (e.g., group of both light and dark gray in Figure 2(d)) should be *intersected* with the set of existing cells (e.g., black in Figure 2(a)) to get the actual cells belonging to each group (e.g., only dark gray in Figure 2(d)).

Since the "non-existent" locations are known to both parties, we do not need to take them into account for the leaf value codelength, which is still $m' H(m'_1/m', \ldots, m'_k/m')$ (see Lemma 1), where $m'_p$ is the number of quadtree leaves having value $p$ ($1 \le p \le k$) and $m' = \sum_{p=1}^{k} m'_p$. However, for the tree-structure codelength we need to *include* the number $m'_0$ of nodes corresponding to non-existent locations (e.g., white in Figure 2(c)). Thus, the structure codelength is $\lceil 4(m' + m'_0)/3 \rceil - 1$.

## 4 Spatial bi-grouping

In the previous sections we have gradually introduced the necessary concepts that lead up to our final goal: coming up with a simple but powerful description for binary data, which also incorporates spatial information and which allows us to automatically group both cells as well as features, without any user-specified parameters.

In order to exploit dependencies due to spatial affinity, we can pursue two alternatives:

1. Relax the assumption that the values within each $D_{p,q}$ are independent, thus modifying $L(D|Q_X, Q_Y, k, \ell)$. This amounts to saying that, *given* cells $i_1$ and $i_2$ belong to the same group, then it is more likely that feature $j$ will be present in both cells *if* they are neighbouring.
2. Assign a non-uniform prior to the space of possible groupings, thus modifying $L(Q_X, Q_Y, k, \ell)$. This amounts to saying that two cells $i_1$ and $i_2$ are more likely to belong to the same group, *if* they are neighbouring.

We choose the latter, since our goal is to find cell groups that exhibit spatial coherence. In the former alternative, spatial affinity does not decide how we *form* the groups; it only comes into play after the groupings have been decided. The second alternative fortunately leads to efficient algorithms. Each time we consider changing the group of a cell, we have to examine how this change affects the total cost. As we shall see, this test can be performed very quickly.

In particular, we choose to modify the term $L(Q_X|k)$. Let us assume that the dataset has $m = 16$ cells, forming a $4 \times 4$ square (see Figure 1), and that cells are placed into $k = 3$ groups (light gray, dark gray and black in the figure). Instead of transmitting $Q_X$ as an arbitrary $m$-to-$k$ mapping (see Section 3.2), we can transmit the image of $m = 16$ pixels (cells), each one having one of $k = 3$ values. The length (in bits) of the quadtree for this image is precisely our choice of $L(Q_X|k)$ (compare Lemmas 1 and 3).

By using the quadtree codelength, we essentially penalize cell group region complexity. The number of groups is factored into the cost indirectly, since more groups typically imply higher region complexity.

### 4.1 Intuition

For concreteness, let us consider the case of patch locations and species presence features. The intuitive interpretation of cell and feature groups is the following:

- Row (i.e., cell) groups correspond to "neighbourhoods" or "habitats." Clearly, a habitat should exhibit a "reasonable" degree of spatial coherence.
- Column (i.e., species) groups correspond to "families." For example, a group consisting of "gull and pelican" may correspond to "seabirds," while a group with "eagle and falcon" would correspond to "mountain birds."

The patterns we find essentially summarise species and cells into families and habitats. The summaries are chosen so that the original data are compressed in the best way. Given the simultaneous summaries, we wish to make the intersection of families and habitats as uniform as possible: a particular family should either be mostly present or mostly absent from a particular habitat. This criterion jointly decides the species of a family and the cells of a habitat. However, our quad-tree model complexity favours habitats that are spatially contiguous without overly complicated boundaries.

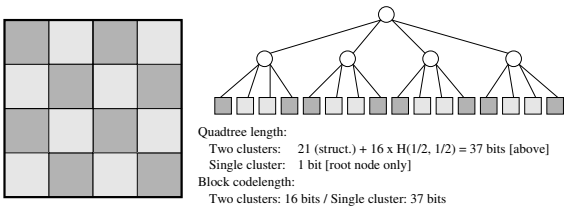



**Figure 3. In this simple example (16 cells and 2 species, i.e., 32 binary values total), if we require groupings to obey spatial affinity, we obtain the shortest description of the dataset (locations and species) if we place all cells in one group. Any further subdivision only *adds* to the total description complexity (due to cell group region shapes).**

The group search algorithms are presented in subsection 4.2. Intuitively, we alternatively re-group cells and features, always reducing the total codelength.

**Example.** A simple example is shown in Figure 3. We choose this example as an extreme case, to clarify the trade-offs between feature and spatial affinity. Experiments based on this boundary case are presented in section 5. Assume we have two species, located on a square map in a checkerboard pattern (i.e., odd cells have only species A and even cells only species B). Consider the two alternatives (we omit the number of bits to transmit species groups, which is the same in both cases):

- Two cell groups, in checkerboard pattern: One group contains only the even cells and the other only the odd cells. In this case, we need 37 bits for the quadtree (see Figure 3). For the submatrices, we need $\lceil \log(8 \cdot 1 + 1) \rceil + 8H(1) = 4$ bits for each of the four blocks (two species groups and two cell groups), for a total of 16 bits. The total codelength is $37 + 16 = 53$ bits.
- One cell group, containing all cells: In this case we need only 1 bit for the (singleton node) quadtree and $\lceil \log(32 \cdot 1 + 1) \rceil + 32H(1/2) = 37$ bits total for the submatrices. The total codelength is $37 + 1 = 38$ bits.

Therefore, our approach prefers to place all cells in one group. The interpretation is that "both species A and B occupy the same locations, with presence in $\rho_{1,1} = 50\%$ of the cells." Indeed, if we chose to perfectly separate the species instead, the cell group boundaries become overly complex without any spatial affinity. Furthermore, if the number of species was different, the tipping point in the trade-off between cell group complexity and species group "impurity" would also change. This is intuitively desirable, since describing exceptions in larger species groups is inherently more complex.

Algorithm INNER:

Start with an arbitrary bi-grouping $(Q_X^0, Q_Y^0)$ of the matrix $D$ into $k$ row groups and $\ell$ column groups. Subsequently, at each iteration $t$ perform the following steps:

1. For this step, we will hold column assignments, i.e., $Q_Y^t$, fixed. We start with $Q_X^{t+1} := Q_X^t$ and, for each row $i, 1 \le i \le n$, we update $Q_X^{t+1}(i) \leftarrow p$, $1 \le p \le k$ so that the choice maximizes the "cost gain"
$$\big(L(D|Q_X^t, Q_Y^t, k, \ell) + L(Q_X^t|k)\big) - \big(L(D|Q_X^{t+1}, Q_Y^t, k, \ell) + L(Q_X^{t+1}|k)\big).$$
We also update the corresponding probabilities $\rho_{p,q}^{t+1}$ after each update to $Q_X^{t+1}$.
2. Similar to step 1, but swapping group labels of columns instead and producing a new bi-grouping $(Q_X^{t+1}, Q_Y^{t+2})$.
3. If there is no decrease in total cost $L(D)$, stop. Otherwise, set $t \leftarrow t+2$, go to step 1, and iterate.

**Figure 4. Row and column grouping, given the number of row and column groups.**

Algorithm OUTER:

Start with $k^0 = \ell^0 = 1$ and at each iteration $T$:

1. Try to increase the number of row groups, holding the number of column groups fixed. We choose to split the row group $p^*$ with maximum per-row entropy, i.e.,
$$p^* := \arg\max_{1 \le p \le k} \sum_{1 \le q \le \ell} |D_{p,q}| H(\rho_{p,q}) / m_p.$$
Construct an grouping $Q_X^{T+1'}$ by moving each row $i$ of the group $p^*$ that will be split ($Q_X^T(i) = p^*$, $1 \le i \le m$) into the new row group $k^{T+1} = k^T + 1$, if and only if this decreases the per-row entropy of group $p^*$.
2. Apply algorithm INNER with initial bi-grouping $(Q_X^{T+1'}, Q_Y^T)$ to find new ones $(Q_X^{T+1}, Q_Y^{T+1})$..
3. If there is no decrease in total cost, stop and return $(k^*, \ell^*) = (k^T, \ell^T)$ with corresponding bi-grouping $(Q_X^T, Q_Y^T)$. Otherwise, set $T \leftarrow T+1$ and continue.

4–6. Similar to steps 1–3, but trying to increase column groups instead.

**Figure 5. Algorithm to find number of row and column groups.**

## 4.2 Algorithms

Finding a global optimum of the total codelength is computationally very expensive. Therefore, we take the usual course of employing a greedy local search (as in, e.g., standard $k$-means [13] or in [4]). At each step we make a local move that always reduces the objective function $L(D)$. The search for cell and feature groups is done in two levels:

- INNER level (Figure 4): We assume that the number of groups (for both cells and features) is *given* and try to find the grouping that minimizes the total codelength. The possible local moves at this level are: (i) swapping feature vectors (i.e., group labels for rows of $D$), and (ii) swapping cell vectors (i.e., group labels for columns of $D$).
- OUTER level (Figure 5): Given a way to optimize for a specific number of groups (i.e., outer level), we progressively try the following local moves: (i) increase the number of cell groups, and (ii) increase the number of feature groups. Each of these moves employs the inner level search.

If $k$ and $\ell$ were known in advance, then one could use only INNER to find the best grouping. These moves guide the search towards a local minimum. In practice, this strategy is very effective. We can also perform a small number of restarts from different points in the search space (e.g., by randomly permuting rows and columns of $D$) and keep the best result, in terms of total codelength $L(D)$.

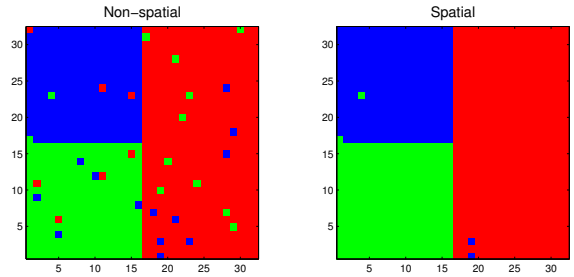
For each row (i.e., cell) swap, we need to evaluate the change in quadtree codelength, which takes $O(\log m)$ time in the worst case (where $m$ is the number of cells). However, in practice, the effects of a single swap in quadtree structure tend to be local.

**Complexity.** Algorithm INNER is linear in the number $nnz$ of non-zeros in $D$. More precisely, the complexity is $O\big((nnz \cdot (k+\ell) + n \log m) \cdot T\big) = O(nnz \cdot (k + \ell + \log m) \cdot T)$, where $T$ is the number of iterations (in practice, about 10–15 iterations suffice). We make the reasonable assumption that $nnz > n + m$. The $n \log m$ term corresponds to the quad-tree update for each row swap. In algorithm OUTER, we increase the total number $k + \ell$ of groups by one at each iteration, so the overall complexity of the search is $O((k^* + \ell^*)^2 nnz + (k^* + \ell^*) n \log m)$, which is is linear with respect to the dominating term, $nnz$.

# 5 Experimental evaluation

In this section we discuss the results our method produces on a number of datasets, both synthetic (to illustrate the intuition) and real. We implemented our algorithms in Matlab 6.5. In order to evaluate the spatial coherence of the cell groups, we plot the spatial extents of each group (e.g., see also [29]). In each case we compare against non-spatial bi-grouping (as presented in Section 3.2). This non-spatial approach produces cell groups of quality similar to or better than, e.g., straight $k$-means (with plain Euclidean distances on the feature bit vectors) which we also tried.

**SaltPepper.** This is essentially the example in Section 4.1, with two features in a chessboard pattern. For the experiment, the map size is $32 \times 32$ cells, so the size




(a) Non-spatial grouping (b) Spatial grouping

**Figure 6. Noisy regions.**

of $D$ is $1024 \times 2$. The spatial approach places all cells in the same group, whereas the non-spatial approach creates two row and two column groups. The total codelengths are (for a detailed explanation, see Section 4.1):

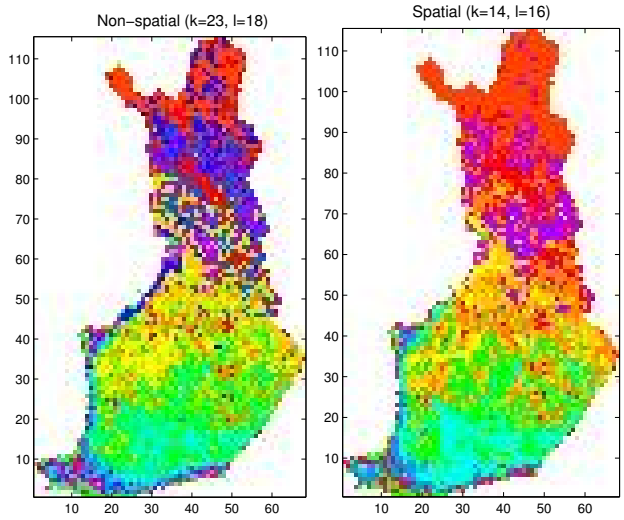
| Groups | Codelength | |
|---|---|---|
| | Non-spatial | Spatial |
| $1 \times 1$ | 2048 + 22 = 2070 | 2048 + 14 = 2062 |
| $2 \times 2$ | 0 + 61 = 61 | 0 + 2431 = 2431 |

**NoisyRegions.** This dataset consists of three features (say, species) on a $32 \times 32$ grid, so the size of $D$ is $1024 \times 3$. The grid is divided into three rectangles. Intuitively, each rectangle is a habitat that contains mostly one of the three species. However, some of the cells contain "stray species" in the following way: at 3% of the cells chosen at random, we placed a wrong, randomly chosen species. Figure 6 shows the groupings of each approach. The spatial approach favours more spatially coherent cell groups, even though they may contain some of the stray species, because that reduces the total codelength. Thus, it captures the "true habitats" almost perfectly (except for a few cells, since the algorithms find a local minimum of the codelength).

**Birds.** This dataset consists of presence information for 219 Finnish bird species over 3813 10Km×10Km patches which cover Finland. The $3813 \times 219$ binary matrix contains 33.8% non-zeros (281,953 entries out of 835,047).

First, the cell groups in Figure 7(b) clearly exhibit a higher degree of spatial affinity than those in Figure 7(a). In fact, the grouping in Figure 7(b) captures the boreal vegetation zones in Finland: the light blue and green regions correspond to the south boreal, yellow to the mid boreal and red to the north boreal vegetation zone.

With resepct to the species groups, the method successfully captures statistical outliers and biases in the data. For example, *osprey* is placed in a singleton group. The data for this species was received from a special study, where a big effort was made to seek nests. Similarly, *black-throared diver* is placed in a singleton group, most likely because of its good detectability from large distances. *Rustic bunting* has highly specialized habitat requirements (mire forests) and is also not grouped with any other species.




(a) Non-spatial grouping ($k = 23$ cell groups, $\ell = 18$ species groups)

(b) Spatial grouping ($k = 14$ cell groups, $\ell = 16$ species groups)

**Figure 7. Finnish bird habitats; our approach produces much more spatially coherent cell groups (see, e.g., red, purple and light blue) and captures the boreal vegetation zones.**

## 6 Related work

In "traditional" clustering we seek to group only the rows of $D$, typically based on some notion of distance or similarity. The most popular approach is $k$-means (see, e.g., [13]). There are several interesting variants, which aim at improving clustering quality (e.g., $k$-harmonic means [30] and spherical $k$-means [7]) or determining $k$ based on some criterion (e.g., X-means [23] and G-means [10]). Besides these, there are many other recent clustering algorithms that use an altogether different approach, e.g., CURE [9], BIRCH [31], Chameleon [16] and DENCLUE [14] (see also [11]). The LIMBO algorithm [2] uses a related, information theoretic approach for clustering categorical data.

The problem of finding spatially coherent groupings is related to image segmentation; see, e.g., [29]. Other more general models and techniques that could be adapted to this problem are, e.g., [3, 19, 24]. However, all deal only with spatial correlations and cannot be directly used for simultaneously discovering feature co-occurrences.

Prevailing graph partitioning methods are METIS [17] and spectral partitioning [22]. Related is also the work on conjunctive clustering [21] and community detection [25]. However, these techniques also require some user-specified parameters and, more importantly, do not deal with spatial data. Information theoretic coclustering [6] is related, but focuses on lossy compression of contingency tables, with distortion implicitly specified by providing the number of row and column clusters. In contrast, we employ MDL and a lossless compression scheme for binary matrices which also incorporates spatial information. The more

recent work on cross-associations [4] is also parameter-free, but it cannot handle spatial information. Finally, Keogh et al. [18] propose parameter-free methods for classic data mining tasks (i.e., clustering, anomaly detection, classification) based on standard compression tools.

Frequent itemset mining brought a revolution [1] with a lot of follow-up work [11, 12]. These techniques have also been extended for mining *spatial collocation patterns* [20, 27, 32, 15]. However, all these approaches require the user to specify a support and/or other parameters (e.g., significance, confidence, etc).

## 7 Conclusion

We propose a method to automatically discover spatial correlation and feature co-occurrence patterns. In particular:

- We group cells and features *simultaneously*: feature co-occurrence patterns help us compress spatial correlation patterns better, and vice versa.
- For cell groups (i.e., spatial correlation patterns), we propose a practical method to incorporate and exploit spatial affinity, in a natural and principled way.
- We employ MDL to discover the groupings and the number of groups, directly from the data, without any user parameters.

Our method easily extends to other natural spatial hierarchies, when available (e.g., city block, neighbourhood, city, county, state, country), as well as to categorical feature values. Finally, we employ fast algorithms that are practically linear in the number of non-zero entries.

## References


[1] R. Agrawal and R. Srikant. Fast algorithms for mining association rules in large databases. In *VLDB*, 1994.

[2] P. Andritsos, P. Tsaparas, R. Miller, and K. Sevcik. LIMBO: Scalable clustering for categorical data. In *EDBT*, 2004.

[3] S. Basu, M. Bilenko, and R. J. Mooney. A probabilistic framework for semi-supervised clustering. In *KDD*, 2004.

[4] D. Chakrabarti, S. Papadimitriou, D. Modha, and C. Faloutsos. Fully automatic cross-associations. In *KDD*, 2004.

[5] T. M. Cover and J. A. Thomas. *Elements of Information Theory*. Wiley-Interscience, 1991.

[6] I. S. Dhillon, S. Mallela, and D. S. Modha. Information-theoretic co-clustering. In *KDD*, 2003.

[7] I. S. Dhillon and D. S. Modha. Concept decompositions for large sparse text data using clustering. *Mach. Learning*, 42, 2001.

[8] P. Grünwald. A tutorial introduction to the minimum description length principle. In *Advances in Minimum Description Length: Theory and Applications*. MIT Press, 2005.

[9] S. Guha, R. Rastogi, and K. Shim. CURE: an efficient clustering algorithm for large databases. In *SIGMOD*, 1998.

[10] G. Hamerly and C. Elkan. Learning the $k$ in $k$-means. In *NIPS*, 2003.

[11] J. Han and M. Kamber. *Data Mining: Concepts and Techniques*. Morgan Kaufmann, 2000.

[12] J. Han, J. Pei, Y. Yin, and R. Mao. Mining frequent patterns without candidate generation: A frequent-pattern tree approach. *Data Min. Knowl. Discov.*, 8(1):53–87, 2004.

[13] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, 2001.

[14] A. Hinneburg and D. A. Keim. An efficient approach to clustering in large multimedia databases with noise. In *KDD*, 1998.

[15] Y. Huang, H. Xiong, S. Shekhar, and J. Pei. Mining confident co-location rules without a support threshold. In *SAC*, 2003.

[16] G. Karypis, E.-H. Han, and V. Kumar. Chameleon: Hierarchical clustering using dynamic modeling. *IEEE Computer*, 32(8), 1999.

[17] G. Karypis and V. Kumar. Multilevel algorithms for multi-constraint graph partitioning. In *SC98*, 1998.

[18] E. Keogh, S. Lonardi, and C. A. Ratanamahatana. Towards parameter-free data mining. In *KDD*, 2004.

[19] J. Kleinberg and E. Tardos. Approximation algorithms for classification problems with pairwise relationships: Metric labeling and Markov random fields. *JACM*, 49(5):616–639, 2002.

[20] A. Leino, H. Mannila, and R. L. Pitkänen. Rule discovery and probabilistic modeling for onomastic data. In *PKDD*, 2003.

[21] N. Mishra, D. Ron, and R. Swaminathan. On finding large conjunctive clusters. In *COLT*, 2003.

[22] A. Y. Ng, M. I. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. In *NIPS*, 2001.

[23] D. Pelleg and A. Moore. X-means: Extending K-means with efficient estimation of the number of clusters. In *ICML*, pages 727–734, 2000.

[24] R. B. Potts. Some generalized order-disorder transformations. *Proc. Camb. Phil. Soc.*, 48:106, 1952.

[25] P. K. Reddy and M. Kitsuregawa. An approach to relate the web communities through bipartite graphs. In *WISE*, 2001.

[26] J. Rissanen and G. G. Langdon Jr. Arithmetic coding. *IBM J. Res. Dev.*, 23:149–162, 1979.

[27] M. Salmenkivi. Evaluating attraction in spatial point patterns with an application in the field of cultural history. In *ICDM*, 2004.

[28] D. J. Vaisey and A. Gersho. Variable block-size image coding. In *ICASSP*, 1987.

[29] R. Zabih and V. Kolmogorov. Spatially coherent clustering with graph cuts. In *CVPR*, 2004.

[30] B. Zhang, M. Hsu, and U. Dayal. K-harmonic means—a spatial clustering algorithm with boosting. In *TSDM*, 2000.

[31] T. Zhang, R. Ramakrishnan, and M. Livny. BIRCH: An efficient data clustering method for very large databases. In *SIGMOD*, 1996.

[32] X. Zhang, N. Mamoulis, D. W. Cheung, and Y. Shou. Fast mining of spatial collocations. In *KDD*, 2004.